

CALL FLOW OBJECT MODEL IN A SPEECH RECOGNITION SYSTEMRelated Applications

This application claims the benefit under 35 U.S.C. § 119(e) of U.S. Provisional Application No. 60/451,353, filed February 27, 2003 and titled "CALL FLOW OBJECT MODEL IN A SPEECH RECOGNITION SYSTEM," which is hereby incorporated by reference. This application is related to U.S. Application No. 10/317,837, filed December 10, 2002 and titled "SPEECH RECOGNITION SYSTEM HAVING AN APPLICATION PROGRAM INTERFACE," which is hereby incorporated by reference.

Background of the InventionField of the Invention

The invention relates to speech recognition technology. More particularly, the invention relates to systems and methods for a speech recognition system having a call flow object model.

Description of the Related Technology

Speech recognition, also referred to as voice recognition, generally pertains to the technology for converting voice data to text data. Typically, in speech recognition systems the task of analyzing speech in the form of audio data and converting it to a digital representation of the speech is performed by an element of the system referred to as a speech recognition engine. Traditionally, the speech recognition engine functionality has been implemented as hardware components, or by a combination of hardware components and software modules. More recently, software modules alone perform the functionality of speech recognition engines. The use of software has become ubiquitous in the implementation of speech recognition systems in general and more particularly in speech recognition engines.

Current speech recognition systems require a call flow designer to generate computer code to implement and modify a particular call flow design that is desired by the designer. The writing of program code requires the designer to be very knowledgeable and highly

skilled in computer programming and in call flow design of speech recognition systems. Thus, the number of people available that are qualified to perform the call flow programming is limited, and the cost of designing the call flow is very high.

Therefore, what is needed is a speech recognition system that implements a call flow object model that enables the call flow designer to implement and modify a particular call flow design without the designer having to write program code or have a detailed understanding of the internal workings of the speech recognition system.

Summary of Certain Inventive Aspects

Certain embodiments of the invention include a method of designing a call flow in a speech recognition system, the method comprising organizing and utilizing a multiple question directed dialog in the speech recognition system, organizing and utilizing a natural language directed dialog in the speech recognition system, and organizing and utilizing an overview dialog in the speech recognition system.

Some embodiments include the method wherein the natural language directed dialog uses a primary concept and a secondary concept. Also included is the method wherein the natural language directed dialog uses a primary concept and a secondary concept without requiring a structured response.

Certain embodiments include a speech recognition system having a plurality of modules for implementing a call flow, the speech recognition system comprising at least one global module defining concepts that are available to the caller in a plurality of places in a call flow, and at least one user-defined module that processes at least a portion of the call flow.

Some embodiments include the speech recognition system wherein the user-defined module is a list module, wherein the list module is a standard caller input module. Also included is the speech recognition system wherein the user-defined module is a natural language module having primary and secondary responses to process natural language input from the caller. Further included is the speech recognition system wherein the user-defined module is an overview module that continues after an interruption in the call flow where the interruption occurred. Still further is the speech recognition system wherein the user-defined

module is a multilevel module that allows multiple questions to be asked by a caller in the same module.

Some embodiments of the speech recognition system further comprise an execution chain that provides actions that are performed in response to input by a caller. Also included is the speech recognition system wherein the execution chain is a main execution chain. Further included is the speech recognition system wherein the execution chain is one or more of the following: a no match execution chain, a no input execution chain, an error execution chain, a response execution chain, a confirmation execution chain.

Certain embodiments include a speech recognition system, comprising a plurality of call flow modules, each module comprising execution chains and grammars, and wherein the modules can have one of a plurality of types, the types including multilevel allowing multiple questions to be asked and linking appropriate grammars to questions in multiple execution chains.

Brief Description of the Drawings

The above and other aspects, features and advantages of the invention will be better understood by referring to the following detailed description, which should be read in conjunction with the accompanying drawings. These drawings and the associated description are provided to illustrate certain embodiments of the invention, and not to limit the scope of the invention.

Figure 1 is a top-level diagram of certain embodiments of a speech recognition system configuration in which a call flow object model operates in a speech recognition engine (SRE).

Figure 2 is a diagram of certain embodiments of the speech recognition engine configuration illustrating the connectivity of various SRE components.

Figure 3 is a diagram of certain embodiments of various components and types of the call flow object model.

Figure 4 is a diagram of one example of a multilevel module that can be devised utilizing the call flow object model.

Figure 5 is a flowchart illustrating an example of a call flow design that can be devised utilizing the call flow objects as described above.

Figure 6 is a flowchart illustrating an example of the call flow design for the main menu module referred to in Figure 5.

Figure 7 is a flowchart illustrating an example of the call flow design for the evaluate input block shown in Figure 6.

Figure 8 is a flowchart illustrating an example of the call flow design for the execute result block shown in Figure 6.

Detailed Description of Certain Embodiments

The following detailed description of certain embodiments presents various descriptions of specific embodiments of the present invention. However, the present invention can be embodied in a multitude of different ways. In this description, reference is made to the drawings wherein like parts are designated with like numerals throughout.

In certain embodiments, an automated call engine refers to a computer with a telephony card that answers incoming calls. A directed dialog may refer to an element of a call flow. Typically the caller is asked a question or presented a list of choice. The caller typically responds by repeating back the choice to the automated call engine. The call flow may refer to the tree of directed dialogs that a caller moves through on the automated call engine. Natural language may refer to the ability of a caller to respond to a directed dialog by using a sentence that must be parsed and interpreted.

In some embodiments, C++ object templates represent elements of a speech recognition system call flow. The way in which the elements are organized and executed allows unique methods of achieving, for example, a multiple question, overview and natural language directed dialog behavior. In addition, the multiple level question dialog allows multiple related questions to be asked of the caller without defining separate directed dialogs for each possibility.

Certain embodiments of an overview dialog play messages to the caller giving some form of overview or summary. The caller may ask a question and the system replies and continues giving the summary where it left off.

Embodiments of the natural language dialog allow primary and secondary concepts to be defined. An example of a primary concept is locations “downtown” and “north county”. The secondary concepts for each could be, for example, “directions” and “store hours”. A caller may state “I would like the store hours for the downtown location.” The recognition engine picks out the key words and applies a score to each primary and secondary concept pair defined. The pair with the highest score may be given as the caller response.

Referring now to the figures, Figure 1 is a top-level diagram of certain embodiments of a speech recognition system 170 configuration in which a call flow object model operates in a speech recognition engine (SRE). In these embodiments, the speech recognition system 170 includes an application 184, which may be one or more modules that customize the speech recognition system 170 for a particular application or use. The application 184 can be included with the speech recognition system 170 or can be separate from the speech recognition system 170 and developed and provided by the user or programmer of the speech recognition system 170. In certain embodiments, the call flow object model is incorporated into the one or more modules of the application 184. In other embodiments, the call flow object model may be implemented by incorporating it into other modules or applications.

In this embodiment, the speech recognition system 170 includes input/output audio sources, shown in Figure 1 as a source 1 input/output 174 and a source 2 input/output 178. While two audio sources are shown in Figure 1, the speech recognition system 170 may have one or a multiplicity of input/output audio sources. In addition, the audio source may be of various types, e.g., a personal computer (PC) audio source card, a public switched telephone network (PSTN), integrated services digital network (ISDN), fiber distributed data interface (FDDI), or other audio input/output source. Some embodiments of the speech recognition system 170 also include a database of application specifications 180 for storing, for example, grammar, concept, phrase format, vocabulary, and decode information. The objects and other data items saved by and comprising the call flow object model may be stored within the database of application specifications 180, or alternatively may be stored in other storage devices such as electronic memory devices, hard disks, floppy disks, CD-ROMs, DVDs, or the like.

The speech recognition system 170 additionally includes a speech recognition engine (SRE) 190. The functions of the SRE include processing spoken input and translating it into a form that the system understands. The application 184 can then either interpret the result of the recognition as a command or handle the recognized audio information. The speech recognition system 170 additionally includes a speech recognition engine application program interface (API) 194, or speech port API, to enable the programmers or users to easily interact with the speech recognition engine 190.

Figure 2 is a diagram of certain embodiments of the speech recognition engine 190 configuration illustrating the connectivity of various SRE components. The application 184 is shown in Figure 2 as an oval to illustrate that in this embodiment the application 184 is not integral to the SRE 190 but is developed and provided by the user of the system 170. In this embodiment, the user-developed application 184 interacts with the speech port API 194. The speech port API 194 interacts with a word tester module 282 as illustrated by an arrow 280 in Figure 2, e.g., for invoking the speech recognition engine for questions and answers (Q&A) on the recognition session. The speech port API 194 interacts with the speech recognition engine module 190, e.g., for communicating a request to decode audio data as illustrated by an arrow 290 in Figure 2, and for receiving an answer to the decode request as illustrated by an arrow 292.

The word tester module 282 also interacts with a tuner module 286, e.g., for receiving from the tuner module 286 information regarding a recognition session as illustrated by an arrow 284. The tuner module 286 additionally receives from the speech recognition engine 190 information regarding the disk decode request and result files as illustrated by an arrow 288. The tuner 286 interacts with a training program module 294, e.g., for communicating the transcribed audio data to the training program 294 as illustrated by an arrow 298 in Figure 2. The training program 294 also interacts with the speech recognition engine 190, e.g., transferring the new acoustic model information to the speech recognition engine 190 as indicated by an arrow 296.

Figure 3 is a diagram of certain embodiments of various components and types of the call flow object model. A project object 386 is the container for the call flow elements. Each project can contain one or more global modules and one or more user defined modules. In

addition, an audio library and properties may be contained in the project object 386. A module object 388 represents the elements needed to resolve a specific part of a conversation.

The module object 388 can contain one or more execution chain objects 396 and additionally may contain one or more grammar objects 390 as shown in Figure 3. Each grammar object 390 can contain the phrases 394 to handle the predicted responses from the caller. A grammar object 390 can contain one or more vocabulary items. A vocabulary item represents one or more possible responses from a caller. A vocabulary item may have one primary concept and may have one or more secondary concepts.

Each concept 392 represents the core idea a caller is trying to convey through speaking or pressing the touch tone or dual tone multi-frequency (DTMF) buttons. A concept 392 can contain one or more phrases 394. A phrase 394 is the predicted input of a caller. This can be represented by speech, text, numbers or DTMF (numbers). A phrase 394 typically contains no other object.

Each execution chain object 396 can contain one or more actions 398, which are designed to handle a specific event. Each action 398 may have an effect on the call flow the caller is moving through. An action 398 typically contains no other objects. As shown in the embodiment of Figure 3, various actions 398 are possible, for example, vox, transfer, hangup, listen, goto and record. Of course, other desired actions may also be implemented and available for use by the call flow designer.

In some embodiments, the vox action plays audio to a caller. One purpose of the audio is to provide information to the caller or elicit a response from the caller. The goto action directs the call flow to a new module. The listen action indicates that it is time to receive input from the caller and also identifies which grammar to use for interpretation. The transfer action contains the transfer string to move the call out of the system and to a new location (e.g., module). The hangup action marks where to end the call. The record action records a response from a caller.

In certain embodiments, the actions are terminal except the vox action. In other words, a terminal action is the last action in the chain. This is due to the fact that this action performs its function and the call then progresses in a new direction, by entering a new module or execution chain 396. An execution chain object 396 may be a collection of

actions. Each module 388 typically contains at least one execution chain 396 to aid the module 388 in producing a desired response from the caller or deliver some desired information to the caller. In embodiments as shown in Figure 3, the execution chain object 396 has one of a plurality of types, for example, a main type, a no match type, a no input type, an error type, a response type, and a confirmation type. Of course, more or fewer types of execution chains may be implemented, and other types may be implemented that are not shown in the illustrative example of Figure 3.

The main execution chain contains the primary actions to achieve the desired result of the module. Actions allowed include vox, and one other depending on module type. When the main chain fails to elicit a response from the caller, the no input chain's actions can be designed to encourage a response, move the caller along in the call flow or transfer a caller to another agent. Each module can contain one or more no input execution chains. Multiple no input chains may execute sequentially, with each one executing once and the last one executing one or more times. In embodiments with, for example, three no input chains, the first one may be executed the first time the caller does not respond, the second the second time and the third may be executed repeatedly for each no input thereafter. Actions allowed may include, for example, vox, goto, transfer or hang up.

An additional type of execution chain 396 is a no match execution chain. For instance, if the caller responds but the response could not be interpreted, the no match actions can be designed to further clarify how the caller needs to respond. A module can contain one or more no match chains. Multiple no match chains are executed sequentially in the same fashion as the multiple no input chains described above. Actions allowed may include, for example, vox, goto, transfer or hangup.

A further type of execution chain 396 is an error execution chain. If a program error has occurred, the actions in this chain can be designed to be a failsafe. Actions allowed include, for example, vox, goto, transfer or hangup. Another type of execution chain is a response execution chain. Each concept may have a corresponding response chain. The response chain can contain the appropriate actions to perform in response to the callers input. As an example, a caller may say "operator" or press zero. The response chain would contain a transfer action to pass the caller off to the operator. If the caller asks about product

information, the response chain could contain a goto action to move the caller to a module, which would ask, for example, what product they are interested in. Actions allowed may include, for example, vox, goto, listen (in a multilevel module), transfer or hangup.

An additional type of execution chain 396 is a confirmation execution chain. When there is some doubt about the caller's input, a confirmation chain can contain vox actions to elicit a yes/no response to confirm the input. The confirmation chain is considered to have a hidden listen action at the end of the chain. The grammar for this listen action is typically found in the global yes/no module. Actions allowed include, for example, vox.

A phrase 394 can hold text or numbers to represent the possible inputs the caller may make. "Operator" or #0 would be valid examples of caller inputs. A concept 392 can hold multiple phrases that fit under an umbrella identifier. For example, the concept "operator" may contain the phrases "operator", #0 and "front desk". All of these possible inputs would yield the same result and therefore are placed under the concept "operator."

A vocabulary item can contain a primary concept and one or more secondary concepts. In some embodiments, the secondary concepts are only utilized in the natural language module. In other modules that accept input from the caller, the vocabulary item and concept have a one to one relationship. When secondary concepts are present, it is the combination of the primary and secondary concepts that identifies the proper response chain to execute. An example of two vocabulary items each with a primary concept and multiple secondary concepts is as follows:

- Primary: Directions
 - Secondary 1: Downtown
 - Secondary 2: Valley
- Primary: Fax Number
 - Secondary 1: Corporate headquarters
 - Secondary 2: Downtown
 - Secondary 3: Valley

Certain embodiments of the grammar 390 can contain the appropriate responses the caller should make at a given time. The grammar 390 is identified by a listen action. The grammar can contain one or more vocabulary items.

A module 388 is designed to handle one portion of a call flow. For example, this may be a greeting, asking whom the caller would like to speak to, what they would like to know,

asking multiple questions to achieve a specific goal. This may be accomplished by performing various actions at the appropriate times.

The goto module is so named because the main chain's final action can be a goto action. Other valid actions in the main chain are vox actions. This module typically does not receive input from the caller. It is an appropriate place to provide information before moving on in the call flow. Like the goto module, the hangup module gets its name from being the last action in its main chain. In some embodiments, the last action in the transfer module is a transfer action.

In some embodiments, the list module is the standard caller input module. The last action in the main chain of this module is the listen action. The caller is directed to respond in a specific fashion, either by pressing buttons or speaking words. The module is designed with response execution chains to move the caller along in the call flow based on the input the caller provides. The other chains no input, no match and confirmation chains providing support in accomplishing this goal.

The natural language module is typically the same as the list module except that each vocabulary item can have one or more secondary responses to allow more natural input from the caller. Taking the example above:

- Primary: Directions
 - Secondary 1: Downtown
 - Secondary 2: Valley
- Primary: Fax Number
 - Secondary 1: Corporate headquarters
 - Secondary 2: Downtown
 - Secondary 3: Valley

The caller may say, for example: "I need directions to the downtown location." The words "downtown" and "directions" are identified and the appropriate response chain is executed. This naturally assumes that the phrase "downtown" has been specified in the downtown concept and the same is true with the phrase "directions" and its associated concept.

The overview module is typically the same as the list module except that the overview module keeps track of where the caller interrupted a vox action. If the response chain for the input only contains vox actions, the module will pick up where the caller interrupted the vox

The yes/no module is typically the same as the list module except that the yes/no module's main chain's listen action's grammar automatically contains the concepts "yes" and "no". These concepts may pull their phrases for the concepts "yes" and "no" from the global yes/no module.

As an example:

Actions: VOX – “What state do you live in?”
LISTEN – Associates with the grammar containing the 50 United States. Determines user has selected state X.

Actions	VOX – “What city in X do you live in”
	LISTEN – Associates with the grammar containing valid cities in state X. Determines user has selected city Y.

Actions	LISTEN – Associates with a grammar with the concept Start Over
---------	---

Actions VOX – “Here is the information for” $X + Y$

-11-

results in a response X_n + “Start Over” being used when X_n was the input for the first question. Other X concepts will still utilize the response Start Over if that is the input given on the second question.

In certain embodiments, the global yes/no module contains global “yes” and “no” concepts. The phrases defined under these concepts are utilized in the yes/no modules. The confirmation chains utilize this grammar when confirming their input. The main chain may contain a listen action. No goto action may move the caller to this module. The no input and no match chains defined in this module may be utilized as default chains for the confirmation chain and the yes/no modules.

The concepts defined in the global module can be used by other modules. This allows certain concepts to be available to the caller throughout the call flow. The no input and no match chains defined in this module may be utilized as default chains for modules throughout the call flow.

Figure 4 is a diagram of one example of a multilevel module 499 that can be devised utilizing the call flow object model. In the example of Figure 4, multiple execution chains are shown. For example, a main type execution chain 497 is shown, as is a response type execution chain 498. Actions are also shown in conjunction with the execution chains, for example, vox, listen and goto actions. In addition, multiple grammars, vocabulary items, concepts and phrases are illustrated in this example. Of course, Figure 4 shows just one example of a multilevel module that can be devised. Many other examples are also possible depending on the needs and preferences of the designers or users.

Figure 5 is a flowchart illustrating an example of a call flow design that can be devised utilizing the call flow objects as described above. In the example in Figure 5, a start module 592 is devised, in which a main execution chain is executed in block 594, a vox action is processed in block 596, and a goto action is processed in block 598. At block 599, the caller is moved to a main menu module. Certain embodiments of the main menu module are shown in the example of Figure 6 below.

Figure 6 is a flowchart illustrating an example of the call flow design for the main menu module referred to in Figure 5. This example of the main menu module includes the main, response, confirmation, response, no match and no input execution chains. In addition,

the grammar labeled as Grammar 1 is shown having two vocabularies. The two vocabularies are labeled as Vocab 1 and Vocab 2 in Figure 6, and each have a primary concept and several phrases defined. The example in Figure 6 includes an evaluate input block 690, which is illustrated in greater detail below with regard to Figure 7, and an execute result block 694, which is illustrated in greater detail below with regard to Figure 8.

Figure 7 is a flowchart illustrating an example of the call flow design for the evaluate input block 690 shown in Figure 6. Figure 7 shows the detailed operational flow of some embodiments of the evaluate input block 690.

Figure 8 is a flowchart illustrating an example of the call flow design for the execute result block 694 shown in Figure 6. Figure 8 shows the detailed operational flow of some embodiments of the execute result block 694.

While the above detailed description has shown, described, and pointed out novel features of the invention as applied to various embodiments, it will be understood that various omissions, substitutions, and changes in the form and details of the device or process illustrated may be made by those skilled in the art without departing from the intent of the invention.